Web aplikacija za praćenje opreme izdane na korištenje

Hodžić, Adnan

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

Adnan Hodžić

WEB APLIKACIJA ZA PRAĆENJE OPREME IZDANE NA KORIŠTENJE

Diplomski rad

Pula, Svibanj, 2023. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani **Adnan Hodžić**, kandidat za magistra **Informatike** ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

U Puli, 30. Lipnja 2023. godine

Potpis

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, **Adnan Hodžić** dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom **Web aplikacija za praćenje opreme izdane na korištenje** koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 30. Lipnja 2023. godine

Potpis

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Pula, 11. ožujka 2023.

DIPLOMSKI ZADATAK

Pristupnik: **Adnan Hodžić (0303076468)**
Studij: Sveučilišni diplomski studij Informatike

Naslov (hrv.): **Web aplikacija za praćenje opreme izdane na korištenje**
Naslov (eng.): Web application for tracking equipment agreements

Opis zadatka: Zadatak je izraditi web aplikaciju koja bi osobi zaduženoj za izdavanje opreme omogućila jednostavniju obradu zahtjeva za revers (dokument za praćenje internog zaduživanja opreme u firmi) te pojednostavila sami proces generiranja dokumenta i njegovog potpisivanja. Osim obrade zahtjeva aplikacija omogućava praćenje, dodavanje te brisanje opreme i zaposlenika. Aplikacija se dijeli na dva dijela. Prvi dio je korisničko sučelje koje se temeljiti na React.js biblioteci za izradu web aplikacija (SPA) i Bootstrap biblioteci namijenjenoj za brzo stiliziranje komponenti. Drugi dio aplikacije sastoji se od API poslužitelja koji je izgrađen sa Express.js bibliotekom u Node.js okruženju te SQLite ugrađenom relacijskom bazom podataka.
Istražiti prednosti i nedostatke korištenja NoSQL baze u usporedbi s relacijskim bazama.
Opis sustav. Prikazati korisničke scenarije, funkcionalnosti, izraditi potrebne UML dijagrame, opisati implementaciju te na kraju izraditi kratke korisničke upute.

Zadatak uručen pristupniku: 11. ožujka 2023.
Rok za predaju rada: 11. veljače 2024.

Mentor:

_____
Izv.prof.dr.sc. Siniša Sovilj

Sadržaj

1 Uvod

Pratiti opremu unutar firme nije lagan zadatak još ako je uz to potrebno za svakog zaposlenika ručno izraditi ugovor onda posao postaje naporan. Officer aplikacija omogućava jednostavnije kreiranje i potpisivanje ugovora za zaduživanje opreme (revers) tako da osoba zadužena za taj posao jednostavno unese zaposlenike te opremu koju posjeduje kako bi kasnije istu mogla dodijeliti nekom od njih. Za svako zaduženje automatski se generira ugovor u PDF obliku s podatcima od firme, zaposlenika te opreme uz to dodaje se i mjesto namijenjenu potpisu svih odgovornih osoba (npr. Direktor financija, Osoba za upravljanje ljudskim resursima itd.). Svaki zaposlenik može pratiti svoje ugovore te status njihovog potpisivanja

2 Motivacija

2.1 Općenito

Aplikacija je izrađena sa svrhom da olakša manjim i srednjim poduzećima proces prilikom generiranja internih zadužnica za opremu smanjujući tako potrebno vrijeme i resurse tijekom cijelog procesa. Jedan od razloga postojanja ove aplikacije je firma koja je bila u potrazi za raznim aplikacijama koje bi podržavale ovu funkcionalnost. Vecina konkurentnih aplikacija su ogromni ERP sustavi koji zahtijevaju skupu pretplatu te su komplicirani za korištenje ili ne podržavaju digitalno potpisivanje. Neke od tih aplikacija su Konto, softCRM, Relago.

2.2 SWOT analiza

Uz pomoć SWOT analize (slika 1) prikazane su snage, slabosti, prilike te prijetnje aplikacije. Jednostavnost korištenja aplikacije i vrijeme potrebno za obradu procesa svakako predstavljaju snagu aplikacije jer štede korisnikovo vrijeme dok moderan dizaj i cijena dodaju vrijednost i kvalitetu same aplikacije. Neke od slabosti aplikacije bio bi proces instalacije na klijentovoj strani, manjkavost funkcionalnosti te implementacija jednostavnijih sigurnosnih standarda. Sve navedene slabosti aplikacije prilke su koje bi pridonijele većem broju korisnika te mogućnosti povećanja same cijene proizvoda. Svkako konkurencija na tržištu predstavlja prijetnju na način da ponude kvalitetniji proizvod sličnih specifikacija te ponude takav proizvod po puno manjoj cijeni. Jer jedna od ograničenja koju Officer aplikacija ima je što korisit vanjskog poslužitelja o kojoj ovise pojedine funkcionalnosti.

Snage	Slabosti
• Jednostavnost • Efikasnost • Moderan dizajn • Cijena	• Kompliciran proces instalacije • Malo funkcionalnosti • Sigurnosni rizici
Prilike	**Prijetnje**
• Bolja implementacija procesa instalacije • Dodavanje više funkcionalnosti • Bolja implementacija sigurnosti	• Konkurento tržište • Neisplativost

Slika 1: Diagram SWOT analize (Izvor: Autor)

3 Korištene tehnologije

3.1 Node.js

Node.js je ”open-source”, ”server-side” platforma koja omogućuje izvođenjeJavaScript koda na serverskoj strani. Temelji se na V8 ”JavaScript engineu”, kojeg koristi Google Chrome preglednik, te pruža programerima visokokvalitetan i učinkovit način za razvoj web aplikacija. Izvorno razvijen kako bi se olakšao razvoj ”real-time” web aplikacija, poput ”chat-a” i ”online” igara. Glavna prednost Node.js platforme je njezina sposobnost asinkronog programiranja, što znači da aplikacija može izvršavati više funkcija u isto vrijeme, bez zastoja i blokiranja procesa. Ova sposobnost se postiže uporabom programiranja vođeno događajima te povratnih poziva. Node.js također ima bogatu zajednicu razvojnih programera koji stalno dopunjavaju i nadograđuju platformu novim alatima, modulima i funkcijama. Postoje tisuće besplatnih biblioteka i modula dostupnih na npm (Node Package Manager), koji omogućuju programerima brz i jednostavan razvoj web aplikacija. Platforma se može koristiti za razvoj različitih aplikacija, uključujući web servere, aplikacije za razmjenu podataka, RESTful API-je, streaming aplikacije i još mnogo toga. Također omogućuje integraciju s drugim tehnologijama, kao što su MongoDB ili MySQL baze podataka, te omogućuje razvoj skalabilnih i brzih aplikacija.

3.2 React.js

React.js je open-source JavaScript biblioteka koja se koristi za izgradnju korisničkih sučelja za web aplikacije. Razvijen je od strane Facebooka i pruža programerima mogućnost lakog upravljanja korisničkim sučeljem, kao i brz i jednostavan razvoj skalabilnih i performantnih web aplikacija. React.js koristi koncept virtualnog DOM-a (Document Object Model) kako bi se smanjio broj manipulacija sa stvarnim DOM-om, što rezultira bržim performansama aplikacije. React.js omogućuje programerima da opisuju UI komponente kao funkcije, koje se mogu kombinirati kako bi se izgradila složena korisnička sučelja. Također podržava razvoj aplikacija putem *”reusable”* komponenti, što olakšava održavanje i skaliranje aplikacija. Jedna od ključnih značajki React.js-a je jednosmjerno vezivanje podataka (*eng. one-way data binding*), što znači

da promjene u stanju komponente automatski utječu na prikaz korisničkog sučelja. To omogućuje programerima jednostavno upravljanje stanjem aplikacije bez potrebe za ručnim manipuliranjem DOM-om. React.js također ima bogatu zajednicu razvojnih programera koji kontinuirano razvijaju nove alate i module. Postoje tisuće biblioteka i modula dostupnih na npm (*eng. Node Package Manager*), koji pružaju podršku za različite zadatke, poput rutiranja(*eng. Routing*), upravljanja stanjem (*eng. State management*) i još mnogo toga.

3.3 Express.js

Express.js je open-source web aplikacijski okvir za Node.js, koji uvelike olakšava izgradnju web aplikacija i API-ja. Express.js je dizajniran kako bi omogućio programerima brzo i jednostavno upravljanje HTTP zahtjevima i odgovorima. Express.js koristi modularan pristup u kojem se funkcionalnost aplikacije izgrađuje pomoću middleware funkcija koje se izvršavaju nakon dolaska HTTP zahtjeva. Ovo omogućuje programerima da lako dodaju funkcionalnosti poput autentikacije, upravljanja stanjem i rutiranja. Express.js također pruža jednostavan način za povezivanje s bazom podataka, kao što su MongoDB ili MySQL, te omogućuje programerima da koriste različite ORM-ove (Object-Relational Mapping) kako bi se olakšao rad s bazom podataka. Jedna od ključnih prednosti Express.js-a je jednostavnost i fleksibilnost. Express.js se može koristiti za razvoj bilo kakve web aplikacije, od jednostavnog bloga do kompleksnog web sustava. Express.js također ima bogatu zajednicu razvijatelja koji kontinuirano razviju nove alate, module i funkcionalnosti koji pružaju podršku za različite zadatke u razvoju web aplikacija. Uz sve navedeno, Express.js je jedna od najpopularnijih biblioteka za izgradnju web aplikacija u Node.js okruženju, a njegova popularnost i podrška će vjerojatno nastaviti rasti u budućnosti.

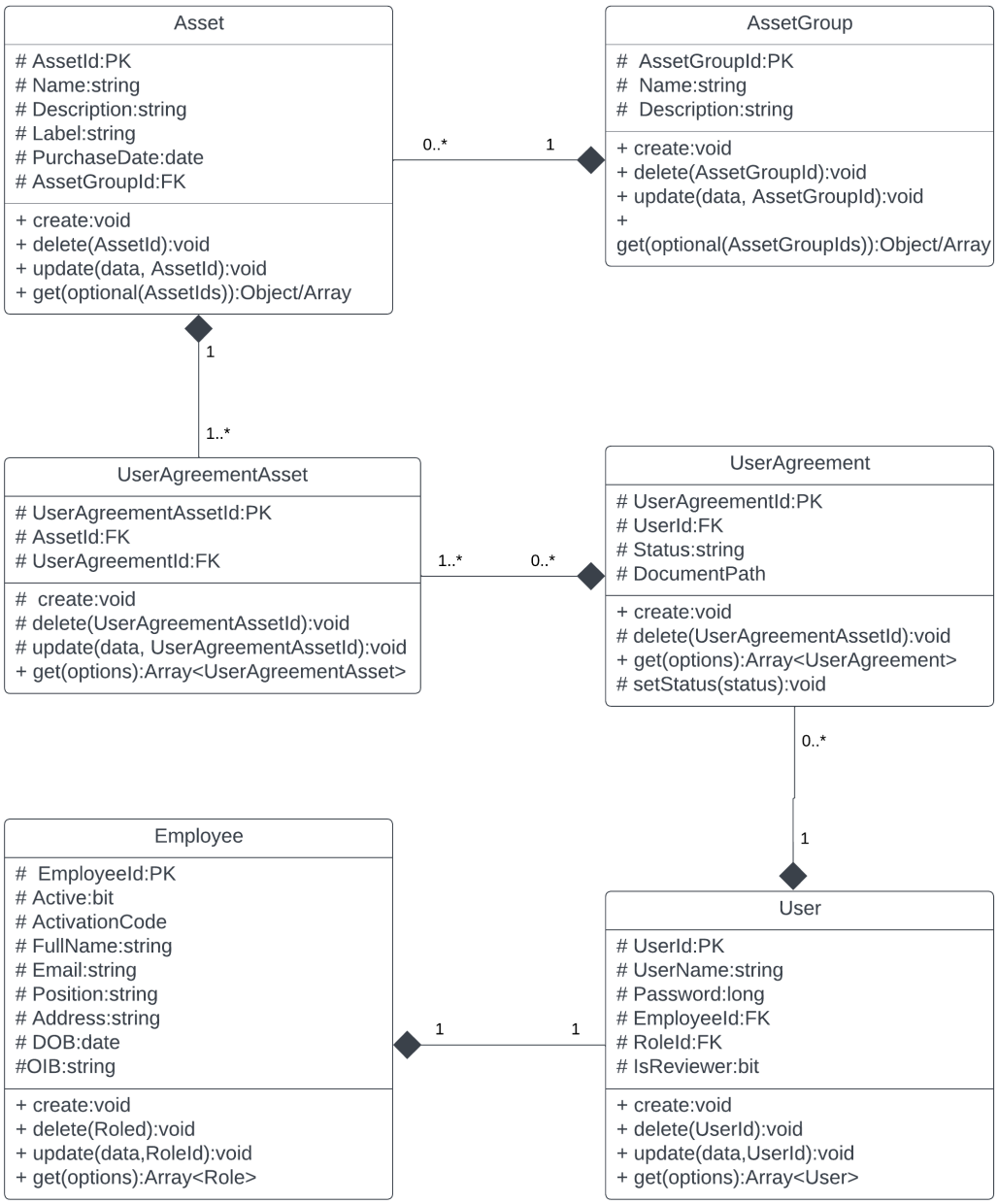
3.4 Sqlite

SQLite je open-source relacijska baza podataka koja je osmišljena za ugrađene sustave i mobilne aplikacije, ali se također može koristiti za izgradnju manjih web aplikacija. SQLite se može smatrati ”lightweight” bazom podataka, jer je jednostavan za korištenje i ne zahtijeva značajne resurse računala ili servera. Jedna od ključnih prednosti SQLite-a je njegova portabilnost. SQLite je neovisan o platformi i može se koristiti na različitim operativnim sustavima, kao što su Windows, Linux i macOS, a može se koristiti i na mobilnim uređajima s iOS i Android operativnim sustavima. SQLite baza podataka se također može lako prenositi, jer se cijela baza podataka nalazi u jednoj datoteci. SQLite podržava standardne SQL upite, uključujući SELECT, INSERT, UPDATE i DELETE, što omogućuje programerima da upravljaju podacima u bazi podataka na uobičajen način. Također podržava transakcije i referencijalni integritet, što povećava pouzdanost baze podataka i osigurava konzistentnost podataka. Postoji puno biblioteka dostupnih na različitim programskim jezicima koje omogućuju programerima da lako povežu svoju aplikaciju sa SQLite bazom podataka. Uz sve navedeno, SQLite se može koristiti za različite vrste aplikacija, od manjih projekata do velikih sustava, jer nudi jednostavnost, brzinu i pouzdanost u radu s bazom podataka.

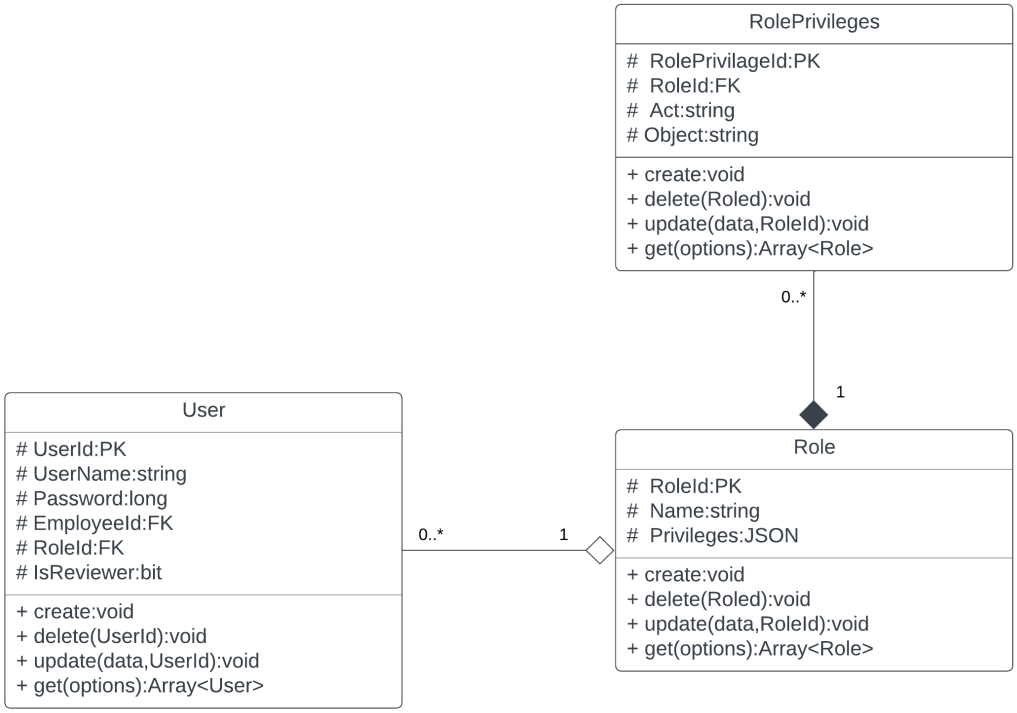
4 Razrada funkcionalnosti

4.1 Klasni UML dijagram

Klasnim dijagramom (slika 2 i slika 3) predstavljani su objekti aplikacije. Iz dijagrama se vidi da ”User” objekt sadrži podatke o korisniku a proširuje ju objekt ”Employee” koji sadrži podatke o zaposleniku poput imena i prezimena, datuma rođenja, OIB-a itd. Prilikom kreiranja zahtjeva za opremom instancira se objekt ”UserAgreement” koji sadrži jednog ”User”-a te jednog ili više ”UserAgreementAsset”-a što predstavlja svu opremu kreiranog zahtjeva. ”UserAgreementAsset” proširen je sa ”Asset” objektom koji sadrži podatke o opremi, a ”Asset” objekt je sadržan u ”AssetGroup” objektu koji grupira opremu. Na slici 3 možemo vidjeti da ”User” objekt proširuje ”Role” objekt koji predstavlja ulogu korisnika u aplikaciji poput zaposlenika, upravitelja ureda ili administratora. ”Role” objekt sadrži ”RolePrivilages” objekt koji opisuje koje radnje korisnik može činiti u aplikaciji poput brisanja, dodavanja i čitanja.



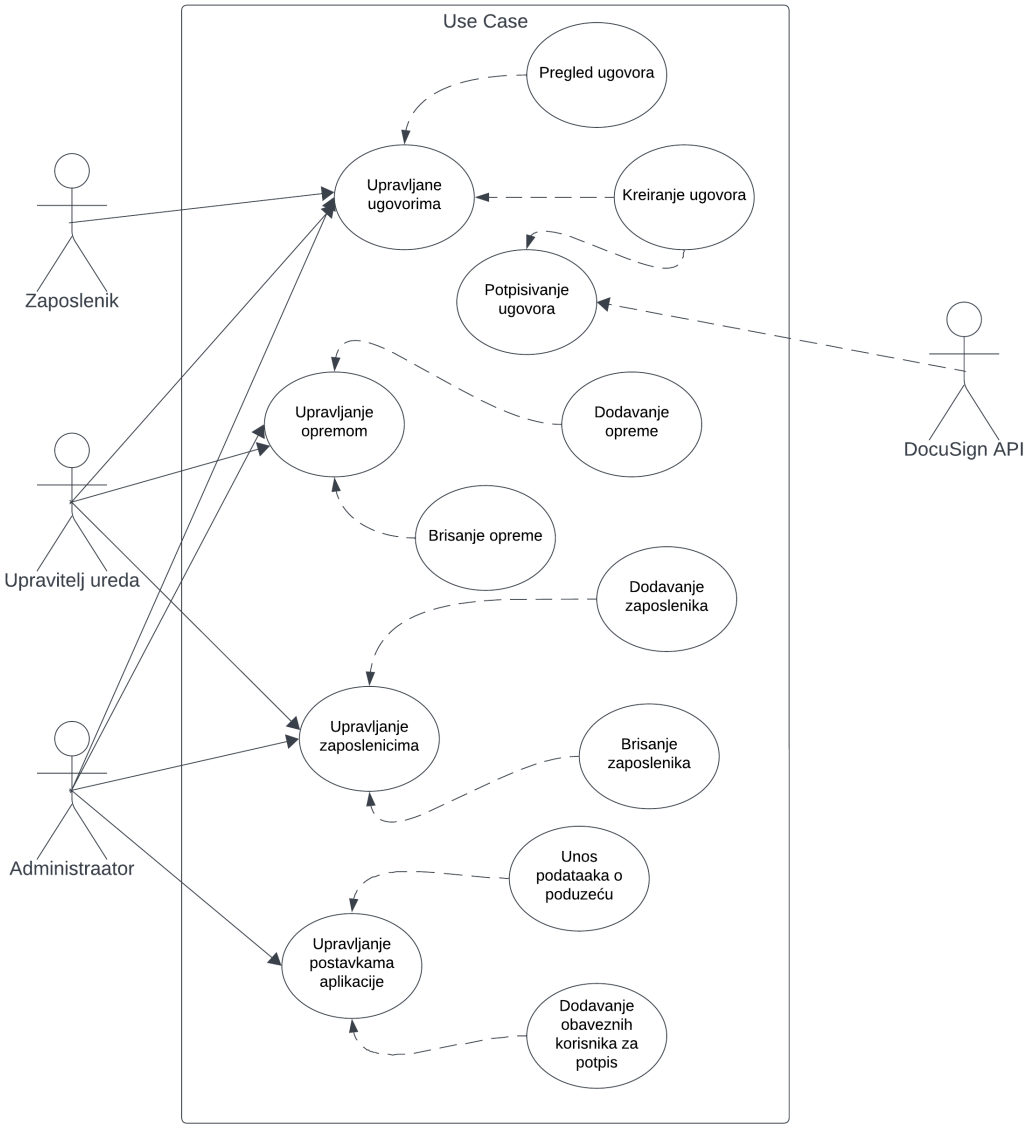
Slika 2: Klasni UML diagram (Izvor: Autor)



Slika 3: Klasni UML diagram - User (Izvor: Autor)

4.2 Use case UML dijagram

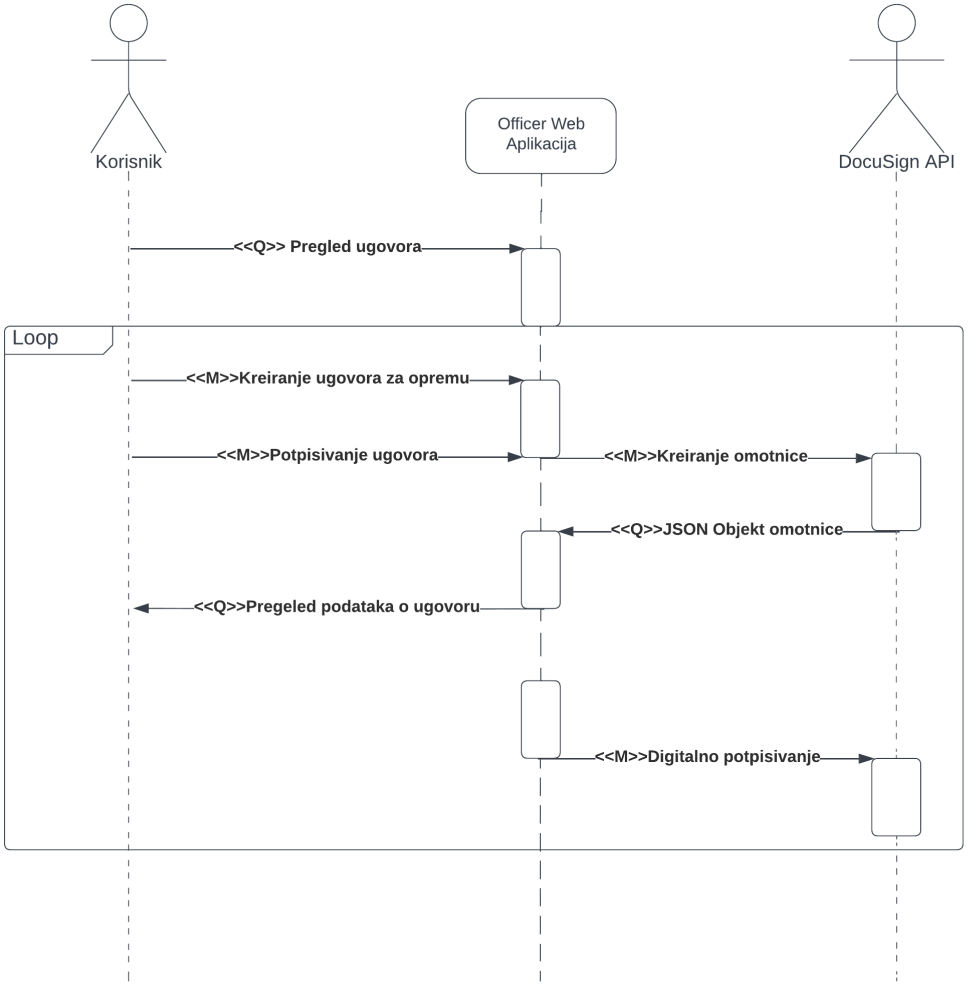
Use case dijagramom (slika 4) prikazane su funkcionalnosti aplikacije te interakcija različitih korisnika i vanjskog klijenta. Iz priloženog, korisnik s pravima zaposlenika ima pravo korištenja funkcionalnosti za kreiranje ugovora, što čini putem izrade zahtjeva za zaduženje opreme. Također, zaposlenik ima uvid u sve svoje ugovore. Upravitelj ureda i Administrator imaju sva prava kao i zaposlenik, s dodatnom mogućnošću upravljanja opremom i zaposlenicima. Administrator također ima dodatna prava za upravljanje postavkama aplikacije, kao što su unos podataka o poduzeću te dodavanje obaveznih korisnika za digitalni potpis ugovora.



Slika 4: Use case UML diagram (Izvor: Autor)

4.3 UML dijagram slijeda

Dijagramom slijeda (slika 5) prikazana je glavna funkcionalnost aplikacije, a to je izrada ugovora o zaduženju opreme (slika 6). Korisnik priliko ulaska u aplikaciju sa svojim vjerodavnicama može vidjeti prikaz sa svim svojim dokumentima. Odabirom prikaza za izradu novog ugovora otvara se forma koju je obavezno ispuniti te podnijeti. Nakon podnošenja izrađuje se objekt u bazi te se šalje zahtjev na vanjskog poslužitelja za digitalno potpisivanje. Korisnik može odabrati da ugovor potpiše odmah ili može odložiti potpisivanje za kasnije.



Slika 5: Sequence UML diagram (Izvor: Autor)

Zadužnica za opremu

UNIPU2, Marka Marulica 12, OIB: 12312312311, (u daljnjem tekstu Poslodavac)

i

Officer Admin iz Rovinj, OIB: 12312312312 (u daljenjm tekstu Radnik)

su u **Puli** dana 12/09/2023 zaključili slijedeće

Potvrda o zaduženju opreme

Radik je primio opremu sa sljedećim specifikacijama

1. Asus Vivobook X16

Officer Admin
Super Admin

Officer Manager
Manager

Slika 6: Primjer generiranog ugovora (Izvor: Autor)

5 Implementacija projekta

5.1 Web aplikacija

5.1.1 Arhitektura

Web aplikacija je razvijena u React.js biblioteci kako bi razvoj bio jednostavniji i efikasniji. Za arhitekturu aplikacije od samog početka razvoja planirano je kreirati jezgrene komponente *(eng. "core components")* poput tablica, modalnog prikaza te komponenti za učitavanje i uređivanje slika. To omogućuje da se tijekom razvoja više koristi konfiguracijski pristup nego direktno kodiranje ili u još gorem slučaju ponavljanje istog koda. Aplikacija je osmišljenja na SPA (eng. *"Single Page Application")* principu što znači da sve kreće iz jednog glavnog prikaza te se s obzirom na promjenu rute URL-a mijenja potprikaz. Svaki potprikaz ujedno obuhvaća jednu od funkcionalnosti aplikacije npr. prikaz za kreiranje ugovora. Svaki od prikaza sadrži jednu ili više komponenti koje su uglavnom izvedene iz jezgrenih komponenti.

5.1.2 Dizajn

Prilikom razvoja dizajna ideja je bila koristiti već predefinirane komponente iz ”boostrap” biblioteke što je za dio komponenti i učinjeno, ali radi postizanja malo ”ozbiljnijeg” izgled ipak je veći dio komponenti prilagođenog dizajna. Jedan od prvih zahtjeva je bilo korištenje pravila ”60-30-10”. To znači korištenje tri boje (slika 7) od kojih je jedna bazna i obuhvaća 60% aplikacije i najčešće se pojavljuje kao podloga bez interaktivnog aspekta. Ostale dvije boje koriste se za naglašavanje interaktivnosti komponenti.

Slika 7: Paleta boja korištena za sučelje web aplikacije (Izvor: Autor)

5.2 Poslužiteljska aplikacija

5.2.1 Autentikacija

Za razvoj poslužiteljske strane aplikacije korišten je Express.js framework iako je mogao poslužiti bilo koji drugi lakši framework poput Flask-a. Na odabir je najviše utjecala iskustvo s JavaScriptom za razliku od ostalih programskih jezika. U početku razvoja aplikacije dosta vremena je posvećeno na implementaciju kvalitetne autentikacije iako i dalje postoji mjesto za napredak koje će biti spomenuto kasnije. Autentikacija između web i poslužiteljske aplikacije vrši se uz pomoć JWT standarda što omogućava da poslužiteljska strana generira token s obzirom na vjerodajnicu poput korisničkog imena i lozinke te s time dobije određena prava prilikom korištenja servisa. Osim autentikacije implementirana je i autorizacija sto znaci da korisnik ima ograničeno korištenje poslužiteljske aplikacije. Prava se konfiguriraju u bazi tako da se korisniku s obzirom na ulogu u aplikaciji dodjeljuju akcije koje on može izvršavati za određene entitete. Jedna od mogućih unaprjeđenja u implementaciji autentikacije je korištenje pristupnog i tokena za osvježavanje te spremanje tokena u kolačić kako bi se spriječili mogući XSS (eng. ”Cross Scripting Attacks”) napadi.

5.2.2 Rutiranje i kontroleri

Svaki entitet u bazi ima svoju rutu za četri osnovne operacije, a to su dodavanje, brisanje, čitanje i ažuriranje. Za to postoji jezgrena komponenta koja se koristi za sve kontrolere (slika 8), ali radi omogućavanja fleksibilnosti svaku od prije navedenih operacija moguće je prepisati i prilagoditi za druge potrebe. Jedan od primjera je brisanje ugovora. To u pravilu ne bi smjelo biti moguće jer bi se time narušila integracija podataka. Jedno od rješenja je takozvano mekano brisanje (eng. soft delete). U tom se slučaju ne briše podatak iz baze nego se simulira brisanje postavljenjem zastavice na tom zapisu,a kasnije se po toj zastavici filtriraju podatci te se ne prikazuju korisniku.

ApplicationSettings > application-settings-controller.js > ...

```
1   const coreController = require("../Core/core-controller");
2   const openConnection = require("../../services/database");
3   exports.get = function(object){ return async (req,res) =>{
4       const { name } = req.query;
5
6       const sql = `SELECT * FROM ${object} WHERE Name=?`;
7       try {
8           const db = await openConnection();
9           const rows = await db.get(sql, [name]);
10          db.close();
11          return res.status(200).json(rows);
12      } catch (err) {
13          console.log(err);
14          return res.status(500).json(err);
15      }
16  }}
17  exports.create = coreController.create
18  exports.update = coreController.update
19  exports.delete = coreController.delete
```

Slika 8: Primjer korištenja jezgrene komponente u kontroleru (Izvor: Autor)

5.2.3 Poslužitelj za digitalno potpisivanje

Digitalno potpisivanje dokumenta obavlja se kroz API servis aplikacije DocuSign, a izvršava se na način da se prilikom generiranja PDF dokumenta ubace mjesta za potpis koja će sadržavati text s jedinstvenom oznakom korisnika npr. korisničko ime, a kasnije pozivanjem API nad tim dokumentom i slanje dodatnih informacija servisu kreirati objekt s jedinstvenim brojem. Kasnije se tom dokumentu može pristupiti s tim brojem te ga korisnik može digitalno potpisati. Kako korisnik ne bi sam morao odlaziti na web stranicu od DocuSign-a web aplikacija uz klik gumba automatski otvara dokument te je korisnik spreman za potpisivanje. Isto kao i kada korisnik potpise dokument stranica se automatski zatvara te se korisnika vraća na Officer web aplikaciju. Autorizacija API servisa za DocuSign izvedena je kroz singletone klasu (slika 9) koja prilikom inicijalnog pokretanja poslužiteljskog servisa instancira objekt. Taj objekt sadrži token koji se dohvati uz pomoć vjerodajnica koje se dobiju prilikom kreiranja DocuSign računa. Kako token koji nam posluzi DocuSign ima vijek trajanja od 1h poslužiteljska aplikacija će 10000 milisekundi prije isteka tokena dohvatiti novi tako da nema čekanje ili još gore grešku koja će javiti da je token istekao.

```
class PrivateDSToken {
    constructor(){
        console.log("User DSToken class to instatiate object")
    }
    async getToken(){
        if(this.token != null && this.isTokenValid()) return  this.token;
        await this.initToken();
        return this.token;
    }

    async initToken(){
        const dsApiClient = new docusign.ApiClient({basePath: dsConfig.dsApiBasePath});

        try{
            const response = await dsApiClient.requestJWTUserToken(dsConfig.integratorKey, dsConfig.userId,['signature impersonation'],dsConfig.privateKey, 3600)
            this.token = response.body.access_token
            this.iat = Date.now();
        }catch(err){
            console.log(err)
        }
    }
    isTokenValid(){
        return this.iat + 350000 > Date.now()
    }
}

module.exports = class DSToken {
    static _instance;

    static instance() {
        if (!this._instance) {
            this._instance = new PrivateDSToken();
            this._instance.getToken();
        }
        return this._instance;
    }
}
```

Slika 9: Implementacija autorizacije prema DocuSign API servisu (Izvor: Autor)

Kreiranje dokumenta u DocuSign aplikaciji vrši se tako da je potrebno kreirati takozvanu omotnicu (eng. envelope.). Omotnica je objekt u koji se ubacuje u dokument (eng. ”document”) objekt s podatcima o nazivu dokumenta, lokaciji PDF datoteke te s još dodatnim objektima koji će predstavljati osobe potrebne za potpisivanje (slika 10) te lokaciju potpisa na dokumentu. Sve to odrađuje se uz pomoć biblioteke namijenjene za komunikaciju te lakše korištenje DocuSign API servisa. Kada je omotnica instanciran šalje se skupa sa tokenom za autorizaciju te aplikacija dobije povratnu informaciju u JSON obliku koji sadrži jedinstveni broj omotnice kao i neke dodate informacije. Jedinstveni broj se sprema u bazu.

```
const docusign = require("docusign-esign");

exports.getSigners = function(signers){
    return signers.map(signer => {
        const signerObj = docusign.Signer.constructFromObject({
            email: signer.Email,
            name: signer.FullName,
            userName: signer.FullName,
            clientUserId: signer._id,
            recipientId: signer._id,
        });

        const signObj = docusign.SignHere.constructFromObject({
            anchorString: signer.Position,
            anchorYOffset: "-40",
            anchorUnits: "pixels",
            anchorXOffset: "0",
        });

        const tabObj = docusign.Tabs.constructFromObject({
            signHereTabs: [signObj],
        });
        signerObj.tabs = tabObj;

        return signerObj
    })
}
```

Slika 10: Metoda za instanciranje objekata koji predstavljaju mjesto potpisa u PDF dokumentu (Izvor: Autor)

5.3 Baza podataka

Radi lakšeg prenošenja aplikacije implementirana je SQLite baza podataka. Implementacija same baze je vrlo jednostavna. Sve sto je potrebno je jedan file koji se nalazi direktno na poslužiteljskom servisu te Sqlite3 biblioteka koja služi sa spajanje i komunikaciju sa bazom (slika 11). Prednost sqlite3 nad sqilte bibliotekom je mogućnost izvršavanja komandi uz korištenje ”Promisa” te korištenje ”Async/await” sintakse. Za manipuliranje bazom koriste se standardni SQL upiti.

```
const sqlite3 = require('sqlite3').verbose();
const { open } = require('sqlite');
async function openConnection(){
    const db = await open({filename: process.env.DB_PATH,driver: sqlite3.Database});
    db.run('PRAGMA foreign_keys = ON;');
    return db
}
module.exports = openConnection
```

Slika 11: Implementacija konekcije sa SQLite bazom (Izvor: Autor)

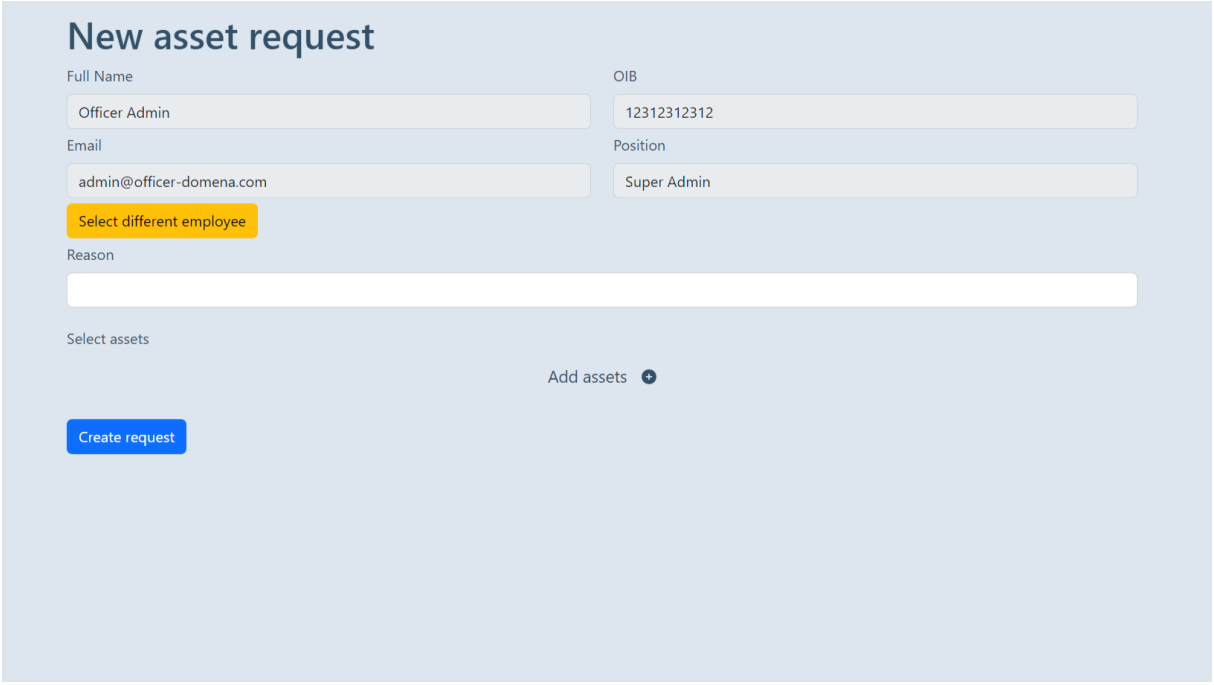
5.3.1 Usporedba SQL i NoSQL baze podataka

SQL i NoSQL baze podataka predstavljaju različite pristupe pohrani i upravljanju podacima. SQL, temeljen na relacijskom modelu, zahtijeva unaprijed definiranu shemu i nudi ACID transakcije, što ga čini prikladnim za aplikacije s čvrstom strukturom podataka i potrebom za transakcijskom konzistencijom. S druge strane, NoSQL baze nude fleksibilnost, skalabilnost i bržu obradu velikih količina podataka zbog svoje ”schema-less” prirode i horizontalne skalabilnosti. Odluka između njih ovisi o specifičnim zahtjevima aplikacije, pri čemu SQL često nalazi primjenu u tradicionalnim aplikacijama, dok NoSQL često dominira u Big Data, e-trgovini i aplikacijama s nestrukturiranim podacima. U današnjem svijetu, često se kombiniraju oba pristupa kako bi se postigao najbolji rezultat za različite aspekte aplikacije.

6 Korisničke upute

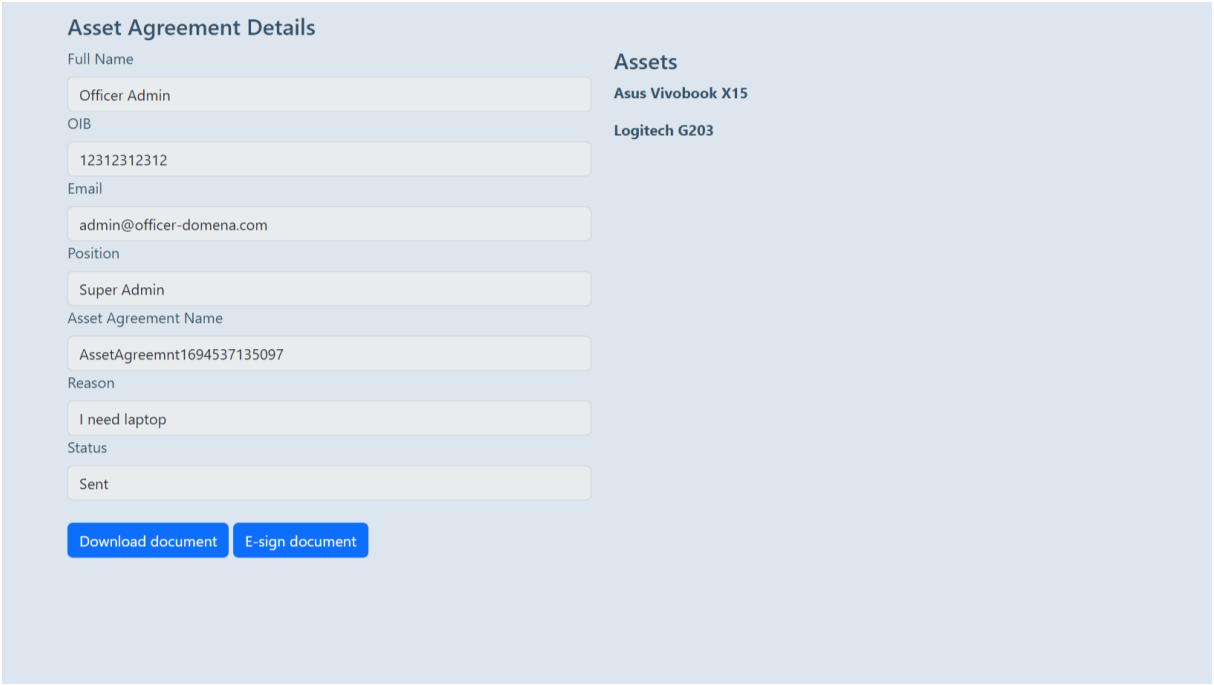
6.1 Izrada reversa za opremu

Prilikom izrade reversa za opremu korisnik odabire sekciju ”New request” iz bočnog izbornika. Klikom na navedenu sekciju otvara se prikaz (slika 12) u kojem korisnik vidi svoje podatke, gumb za odabir drugog korisnika (samo u slučaju ako je prijavljeni korisnik upravitelj ureda ili administrator), ulazno polje za upisivanje razloga zaduženja, sekcija za dodavanje opreme za revers te gumb za slanje zahtjeva.

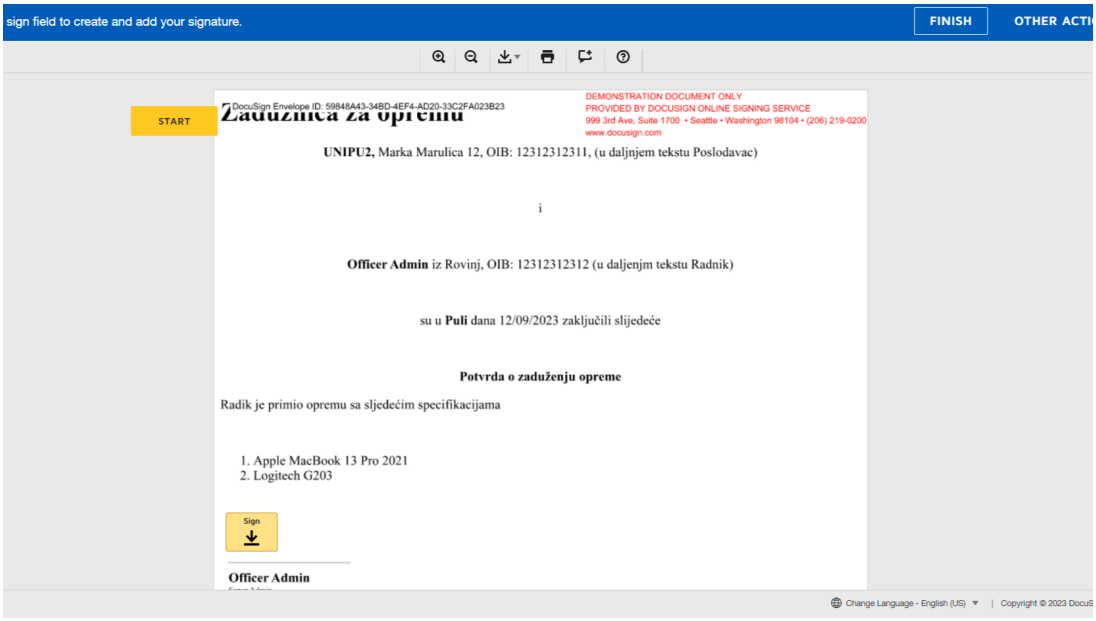


Slika 12: Prikaz sučelja za izradu reversa (Izvor: Autor)

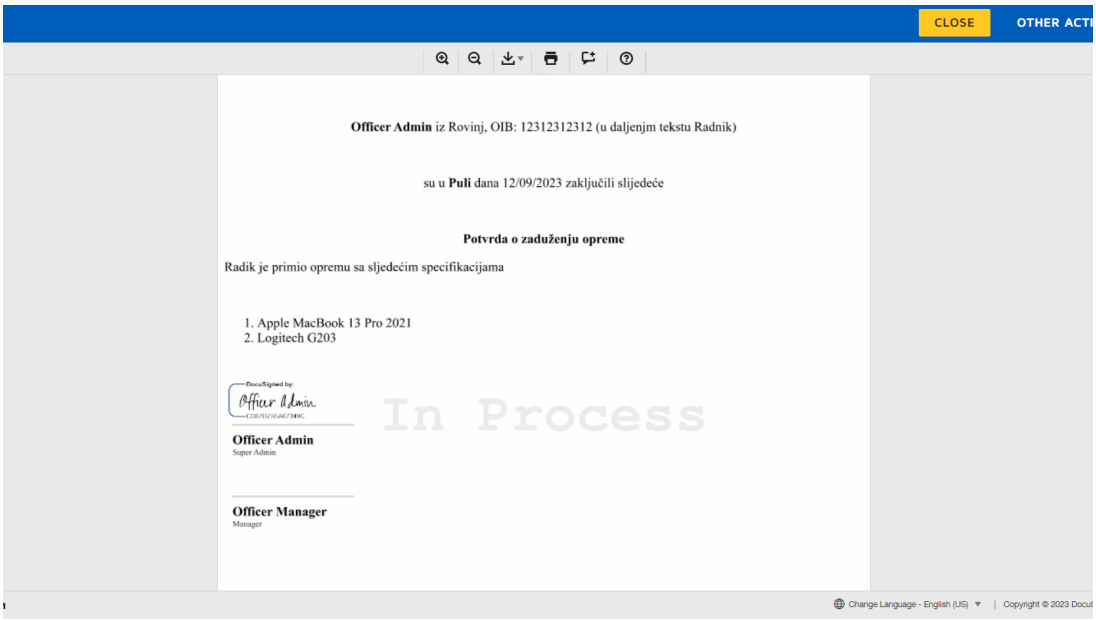
Nakon što je korisnik unio razlog, odabrao barem jedan komad opreme te kliknuo na gumb ”Create request” otvara se detaljni prikaz podataka o reversu (slika 13). U ovom trenutku korisnik ima dvije opcije. Preuzeti revers u PDF obliku klikom na gumb ”Download document” i/ili digitalno potpisati ugovor klikom na gumb ”E-sign document”. Odabirom druge opcije korisniku se otvara DocuSign web aplikacija s prikazom ugovora te mjesta gdje treba digitalno potpisati ugovor (slika 14 i slika 15). Nakon potpisa ugovora aplikacija nas automatski vraća na prethodni prikaz detalja o ugovoru gdje možemo nastaviti koristiti aplikaciju. Kreiranje ugovora svaki korisnik koji je obavezan potpisati ugovor na ”Dashboard” prikazu može vidjeti sve ugovore koje zahtijevaju potpis (slika).



Slika 13: Prikaz detalja reversa (Izvor: Autor)

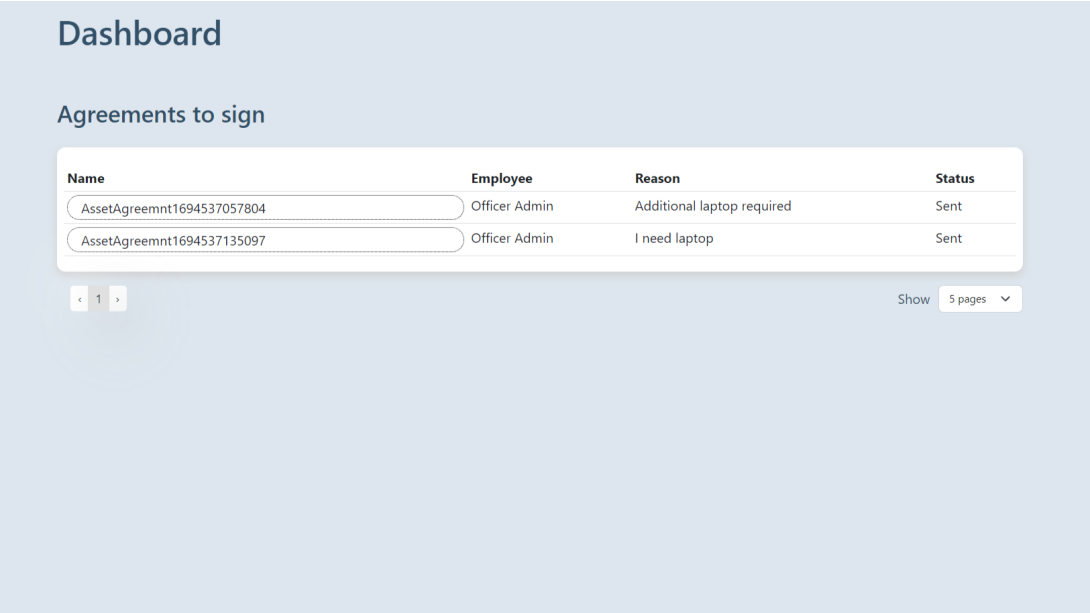


Slika 14: DocuSign prikaz ugovora (Izvor: Autor)



Slika 15: DocuSign prikaz potpisanog ugovora (Izvor: Autor)

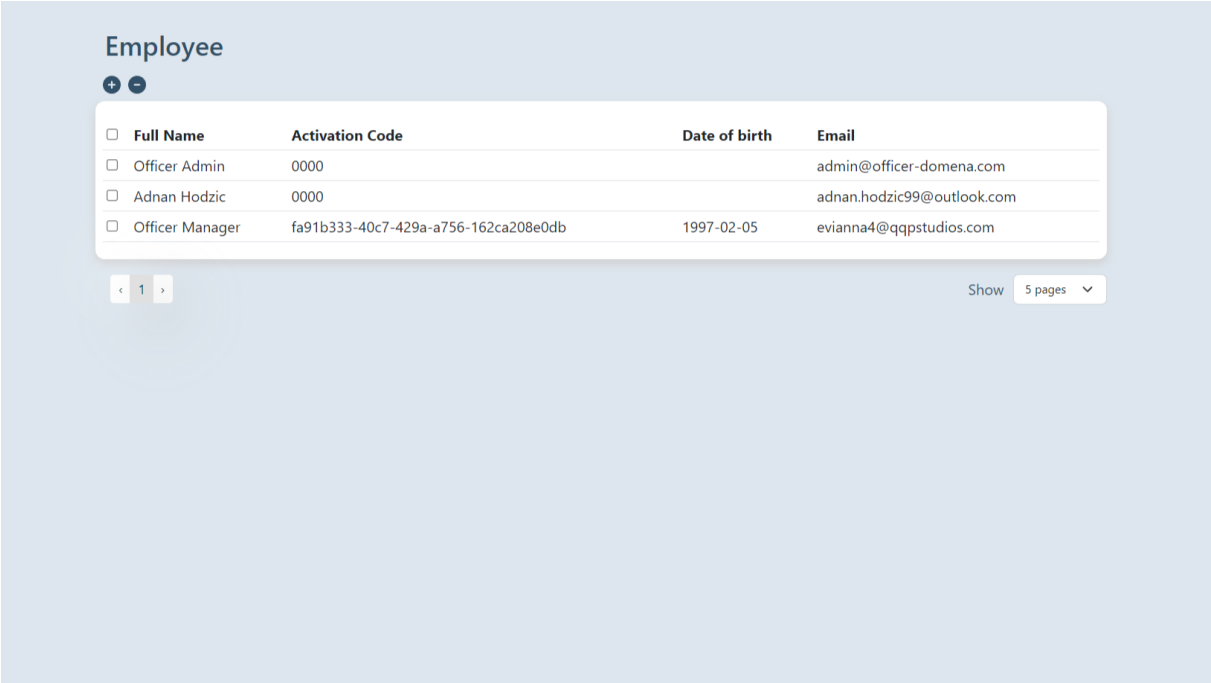
Na ”Dashboard” prikazu korisnik ima mogućnost kliknuti na svaki od nepotpisanih ugovora (slika 16) što će ga odvesti na prikaz detalja od ugovoru gdje ima opciju digitalnog potpisivanja.



Slika 16: Prikaz nepotpisanih ugovora (Izvor: Autor)

6.2 Dodavanje zaposlenika

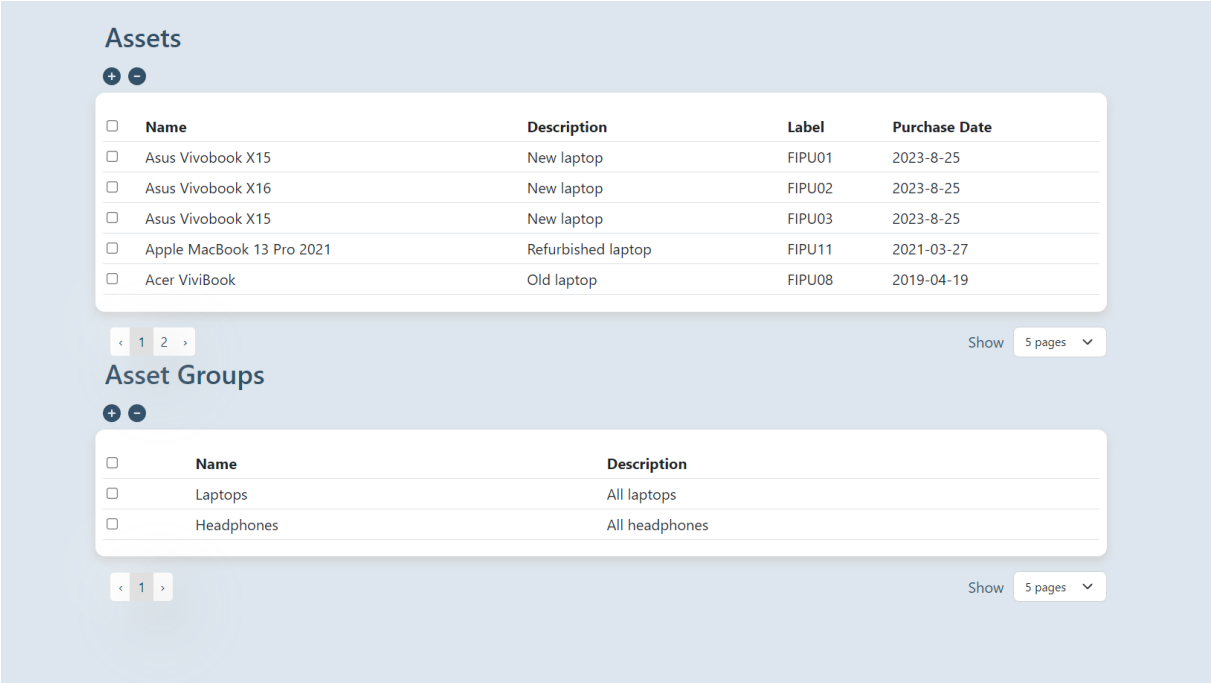
Dodavanje zaposlenika izborom sekcije ”Employee” bočnog izbornika imaju prava samo administrator te upravitelj ureda (slika 17). Prilikom dodavanja korisnika potrebno je unesti podatke poput imena i prezimena zaposlenika, email adresa, datum rođenja, OIB i još detalja o zaposleniku poput pozicije. Kada je zaposlenik kreiran generira se aktivacijski kod (eng. ”Activation Code”) koji je potreban korisniku prilikom registracije u aplikaciju.



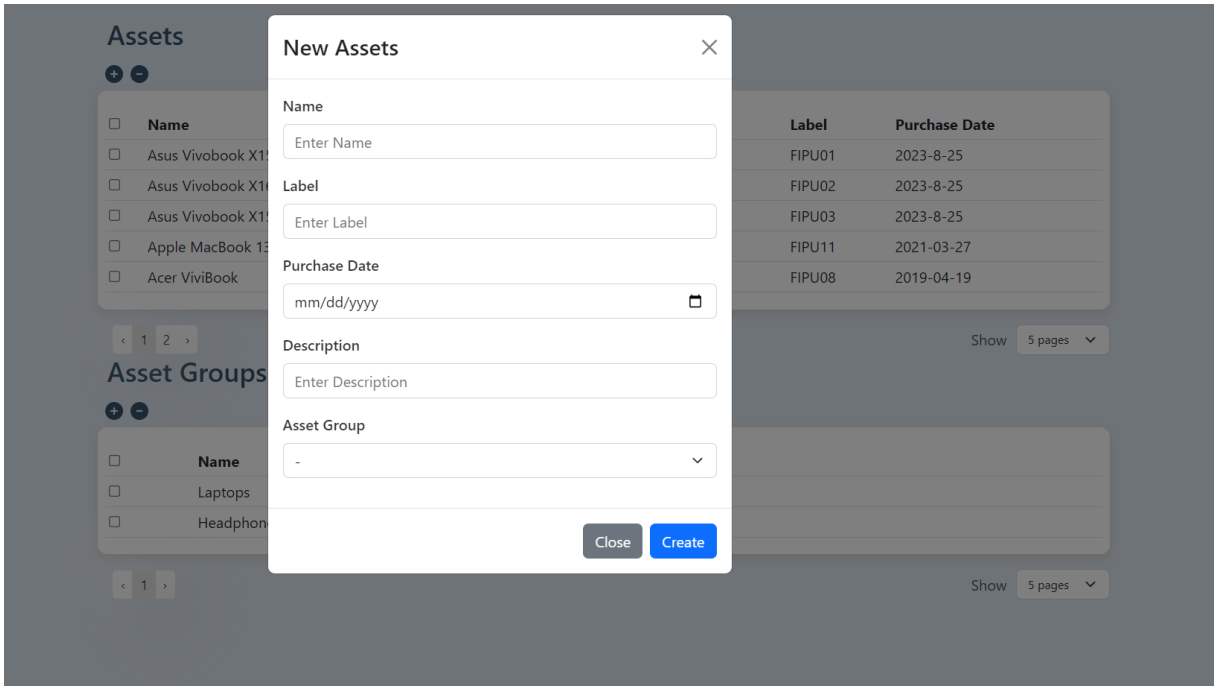
Slika 17: Prikaz sučelja za pregled i upravljanje zaposlenicima (Izvor: Autor)

6.3 Dodavanje opreme

Odabirom ”Assets” sekcije bočnog izbornika otvara se prikaz s pregledom sve dostupne opreme (slika 18). Ova opcija dostupna je samo korisnicima odgovornim za praćenje opreme i administratoru aplikacije. Na prikazu vidimo dvije tablice ”Assets” i ”Asset Groups”. Klikom na ikonu sa znakom plus otvara se modalni prikaz (slika 19) u kojem korisnik ispunjava podatke o opremi poput naziva, opisa, oznake te datuma kada je oprema kupljena.



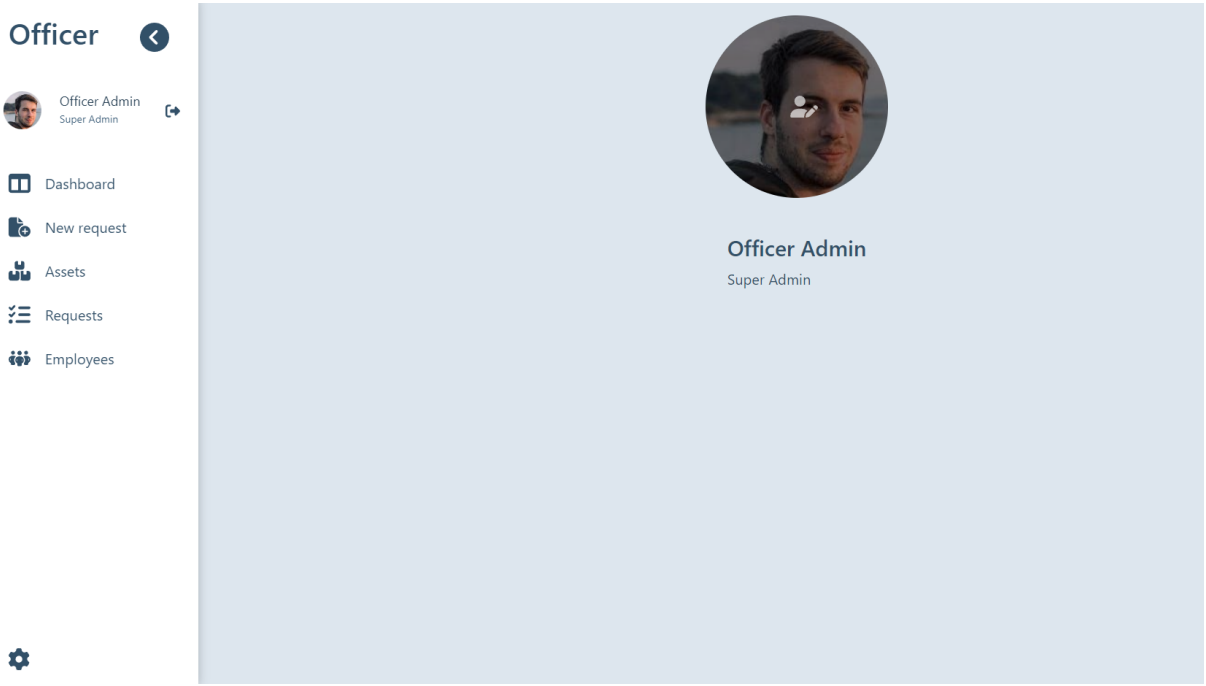
Slika 18: Prikaz sučelja opreme i grupa (Izvor: Autor)



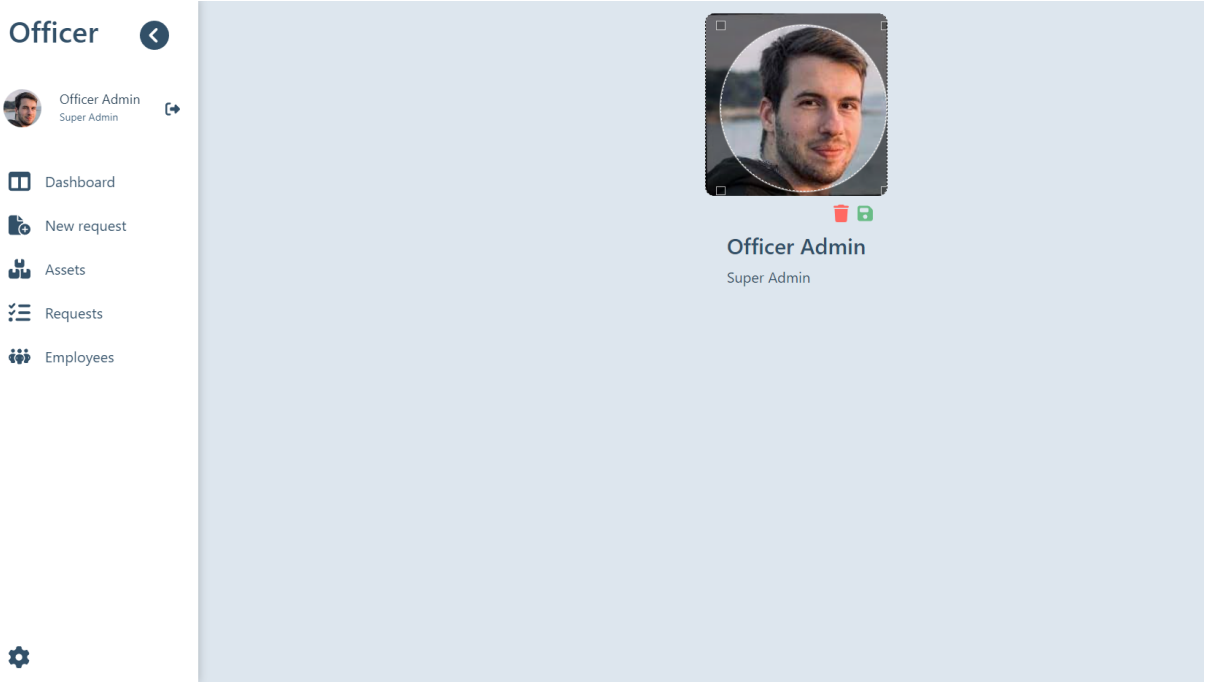
Slika 19: Modalni prikaz za dodavanje opreme (Izvor: Autor)

6.4 Uređivanje profilne slike

Klikom miša na karticu koja se nalazi na vrhu bočnog izbornika otvara se prikaz (slika 20) na kojem možemo vidjeti sliku profila, ako je prethodno dodana, ili mjesto na koje se klikom misa može dodati slika. Klikom na postojeću sliku ili mjesto namijenjeno za dodavanje slike otvara se preglednik datoteka računala u kojem se odabire fotografija koju želimo postaviti. Nakon odabira datoteke imamo mogućnost uređivanja fotografije kao sa slike 21. Klikom zelene ikone odlučujemo spremiti fotografiju dok klikom na crvenu ikonu odbacujemo promjene. Postavke se uređuju klikom na ikonu pored naslova postavki slika i slika.



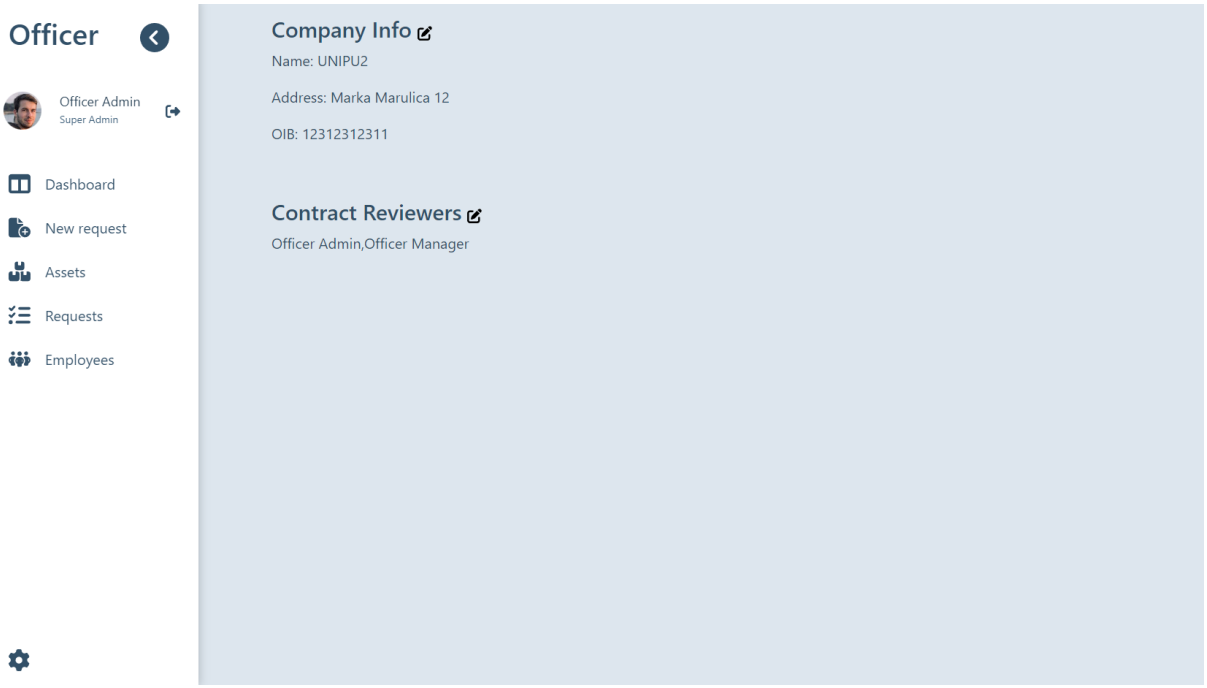
Slika 20: Prikaz sučelja korisničkog profila (Izvor: Autor)



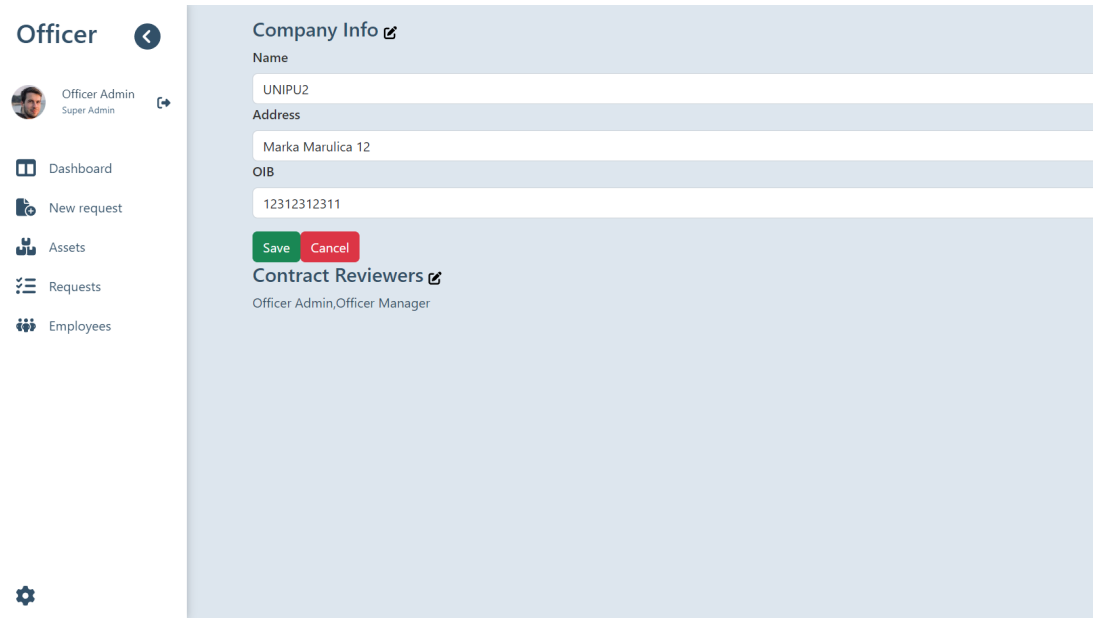
Slika 21: Uređivanje slike profila korisnika (Izvor: Autor)

6.5 Uređivanje postavki aplikacije

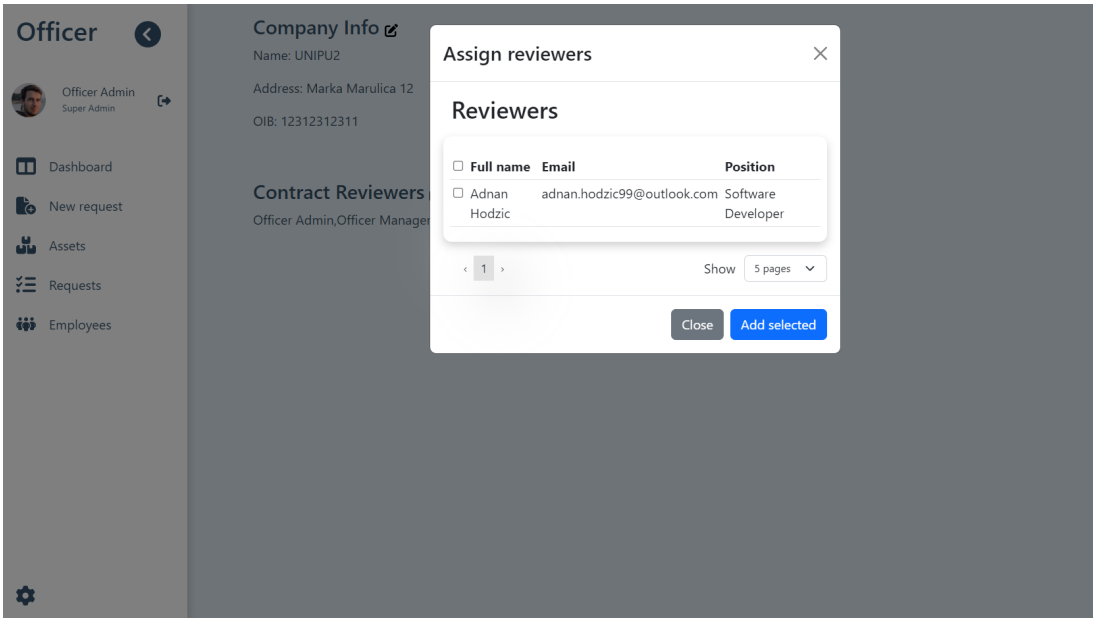
Kroz prikaz postavki aplikacije (slika 22 i slika 23) korisnik sa privilegijama administratora može uređivati podatke o poduzeću poput naziva, adrese te OIB-a. Ovi podatci su bitni jer se nalaze na generiranom PDF ugovoru. Također administrator može odrediti koje osobe moraju pregledati te potpisati svaki od generiranih dokumenata (slika 24). Naravno ako osoba koje izrađuje dokument neće morati potpisivati dva puta ukoliko se već nalazi na listi.



Slika 22: Prikaz za uređivanje postavki aplikacije (Izvor: Autor)



Slika 23: Uređivanje podataka o poduzeću (Izvor: Autor)



Slika 24: Uređivanje liste korisnika zaduženih za potpis ugovora (Izvor: Autor)

7 Zaključak

Prednost Officer aplikacije nad ostalim aplikacija na tržištu definitivno je korištenje aktualnijih tehnologija kako kod poslužiteljskog servisa tako i kod web aplikacije, čineći tako ugodnije iskustvo korisniku priliko korištenja proizvoda zbog modernijeg izgleda ali i razvojinm programerima koji mogu brže i efikasnije razvijati funkcionalnosti. Jednostavnost aplikacije i cijena doprinose kvaliteti prozvoda čineći ga tako dostupnijim manjim i srednjim kompanijama. Aplikacija svakako ima mjesta za napredak dodavanjem dodatnih i unapređenja trenutnih funkcionalnosti.

8 Literatura

Arslan, A. (2023), SQL vs. NoSQL: Choosing the Right Database for System Design, dostupno na: https://medium.com/geekculture/choosing-the-right-database-for-system-design-sql-vs-nosql-and-beyond-d58fde5a6fe3, [27.05.2023]

Lopez, J. (2023), The 60–30–10 Rule: A Foolproof Way to Choose Colors for Your UI Design, dostupno na: https://uxplanet.org/the-60-30-10-rule-a-foolproof-way-to-choose-colors-for-your-ui-design-d15625e56d25, [27.05.2023.]

Mößle, F. (2019), Use HTML and puppeteer to create PDFs in Node.js, dostupno na: https://medium.com/@fmoessle/use-html-and-puppeteer-to-create-pdfs-in-node-js-566dbaf9d9ca, [29.05.2023.]

Paralkar, K. (2022), Singleton Design Pattern – How it Works in JavaScript with Example Code, dostupno na: https://www.freecodecamp.org/news/singleton-design-pattern-with-javascript/, [24.06.2023.]

Sarawgi, S. (2020), Getting Started SQLite3 with Nodejs, dostupno na: https://medium.com/@codesprintpro/getting-started-sqlite3-with-nodejs-8ef387ad31c4, [13.06.2023.]

Wirantono, M. (2020), LocalStorage vs Cookies: All You Need To Know About Storing JWT Tokens Securely in The Front-End, dostupno na: https://dev.to/cotter/localstorage-vs-cookies-all-you-need-to-know-about-storing-jwt-tokens-securely-in-the-front-end-15id, [04.05.2023]

9 Popis slika

10 Prilog

10.1 Repozitorij

Web aplikacija i poslužiteljski servis: https://github.com/adnanh2803/officer-app

11 Sažetak

Ovim diplomskim radom cilj je bio izraditi i opisati web aplikaciju koja bi koristila poduzećima prilikom dodjeljivanje opreme zaposlenicima tako da predajom zahtjeva zaposlenik može zatražiti opremu te bi aplikacija automatizmom generirala PDF dokument koji predstavlja ugovor o zaduženju opreme. Aplikacija također nudi digitalno potpisivanje uz pomoć vanjskog servisa DocuSign. Web aplikacija i poslužiteljski servis razvijeni su uz pomoć aktualnih alata, tehnologija i biblioteka poput Node.js, React.js i Express.js.

Ključne riječi: web aplikacija, ugovor, PDF, DocuSign, Node.js, React.js, Express.js

12 Summary

The aim of this master's thesis was to develop and describe a web application intended for use by businesses in the process of assigning equipment to employees. Through the submission of a request, an employee can request equipment, and the application automatically generates a PDF document representing an asset agreement. Additionally, the application offers digital signing through an external service, DocuSign. The web application and server service were developed using contemporary tools, technologies, and libraries such as Node.js, React.js, and Express.js.

Keywords: web application, agreement, PDF, DocuSign, Node.js, React.js, Express.js